Network Diagnostic Tools

Jijesh Kalliyat
Sr. Technical Account Manager, Red Hat
15th Nov 2014
Network Diagnostic Tools Linux
Tcpdump
Wireshark
Tcpdump Analysis
Sources of Network Issues

If a system is unable to connect to a network,

- Physical layer issues
- Bad network card / drivers or configurations
- Firewall preventing computers from seeing each other

Network Slowness,

- NIC duplex and speed incompatibilities
- Network congestion / Packet drops
- Poor routing
- Bad hardware / cabling
- Overloaded servers
1. Test Network Connectivity

- **Use ping command**
  - Isolate host resolution/DNS issues ( /etc/{hosts,resolv.conf} )
  - Does 127.0.0.1 / local IP/ another host in same network respond?

- **Use traceroute / mtr command**
  - Provides information about path to a remote server.
  - **mtr** : real-time data about latency and routing changes

- **Look for default route / gateway**
  
  `route -n / ip route`

- **Verify the IP address / arp caches**
  
  `ifconfig ( is obsolete ! ) / ip addr list`
  `arp -an / cat /proc/net/arp`
2. Test Remote Ports

➢ **telnet**

```bash
# telnet 192.168.5.5 25
Trying 192.168.5.5...
telnet: Unable to connect to remote host: Connection refused
```

➢ **nc (netcat)**

```bash
# nc example.com 81 -v
nc: connect to node1 port 81 (tcp) failed: Connection refused
# nc example.com 80 -v
Connection to node1 80 port [tcp/http] succeeded!
```

➢ **wget / curl** to check webservers

➢ **nmap** to scan the ports
3. Check the Link status

# dmesg | egrep “eth|em”
e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: None

# ip link show
eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc UP qlen 1000

# mii-tool eth0 ( deprecated, doesn't work on Gigabit NICs )
eth0: negotiated 100baseTx-FD, link ok

# ethtool eth0 ( provided by net-tools )
Speed: 1000Mb/s
Duplex: Half
Link detected: yes

# cat /sys/class/net/eth0/operstate
up
Monitoring and Diagnosing Performance Problems
How Network Interface Cards (NIC) works internally?

- If the NIC is faster than the host, there shouldn't be any problem.
- On the other hand, if the host is faster, then the ring can fill up (no gray area left) and the host will be forced to wait.
Packet Loss at Network Interface Cards (NIC)

- Packet errors/drops displayed in `#ifconfig`

  ```
  eth0   Link encap:Ethernet  HWaddr 00:16:3E:74:7B:63
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:40685771 errors:0 dropped:8 overruns:20 frame:0
         TX packets:2649925 errors:10 dropped:0 overruns:0 carrier:10
  ```

- Firmware Buffer Overflows

  ```
  # ethtool -S

  e1000e/e1000 : rx_missed_errors
  tg3           : rx_discards
  bnx2 driver   : rx_fw_discards
  ```

Increase the buffer size, `ethtool -g eth0 / ethtool -G eth0 rx 4096`
Packet Loss in networking stack (socket / kernel)

- **netstat -s** - Collects infos from the following files: /proc/net/snmp, /proc/net/netstat and /proc/net/sctp/snmp.

  ```
  # netstat -s |egrep -i "error|drop|over"
  7 SYNs to LISTEN sockets dropped
  6 times the listen queue of a socket overflowed
  ```

- **dropwatch** - Monitors and records packets that are dropped by the kernel.

  ```
  # dropwatch -l kas ( /proc/kallsyms are used for function mappings)
  dropwatch> start
  10 drops at unix_stream_sendmsg+735
  5 drops at netlbl_domhsh_def+8349525
  dropwatch> stop
  #
  ```

- **sar -n EDEV** tells you how much errors per second is happening

  ```
  iface     rxerr/s txerr/s  coll/s  rxdrop/s  txdrop/s  txcarr/s  rxfram/s
  eth0      5.00   10.00   0.00    0.00    0.00    5.00      0.00
  ```
TCP Socket Buffers and Tuning

- Flexible buffer that handles incoming and outgoing packets at the kernel level
- Can be tuned in /etc/sysctl.conf file

```plaintext
net.ipv4.tcp_rmem  = 4096 87380 4194304
net.ipv4.tcp_wmem  = 4096 87380 4194304
```

- Be careful not to set the buffers too large. Buffers uses Physical Memory.
- Each time data is read/written to the buffers, the entire socket must be read.
Investigate Sockets

- **ss** - Command-line utility that prints statistical information about sockets
  
  - Shows information similar to 'netstat'
  - Display more TCP and state informations than other tools

```
# ss -t -a  : Display all TCP sockets
# ss -it   : Socket Internal information
```

<table>
<thead>
<tr>
<th>State</th>
<th>Recv-Q</th>
<th>Send-Q</th>
<th>Local Address:Port</th>
<th>Peer Address:Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESTAB</td>
<td>967899</td>
<td>0</td>
<td>192.168.1.2:35390</td>
<td>192.168.1.3:imaps</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>cubic</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>wscale:11,7</td>
<td>rto:204</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>rtt:1.875/1</td>
<td>ato:40</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>mss:1448</td>
<td>cwnd:10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>send 61.8Mbps</td>
<td>rcv_rtt:40.875</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>rcv_space:203340</td>
<td></td>
</tr>
</tbody>
</table>

**Recv-Q**: The count of bytes not copied by the program connected to this socket.

**Send-Q**: The count of bytes not acknowledged by the remote host.
### Other Candidates?

- **Are interrupts balancing?** ( `/proc/interrupts` )
  
<table>
<thead>
<tr>
<th>CPU0</th>
<th>CPU1</th>
<th>CPU2</th>
<th>CPU3</th>
</tr>
</thead>
<tbody>
<tr>
<td>17704</td>
<td>28952</td>
<td>8866</td>
<td>8027</td>
</tr>
</tbody>
</table>
  
  ```
  PCI-MSI-edge eth1
  ```

- **Multiqueue enabled?** ( `/proc/interrupts` )
  
  ```
  PCI-MSI-X eth2-tx-0
  PCI-MSI-X eth2-tx-1
  ```

- **Identify Driver Issues**
  
  ```
  # grep eth var/log/messages & var/log/dmesg
  # grep <driver name> var/log/messages & var/log/dmesg
  ```

- **Check relevant offloading and apply**
  
  ```
  # ethtool -k eth0
  rx-checksumming: off
  tx-checksumming: on
  ```
Where does tcpdump get the packets?

Networking Stack – Receiving Side

User Space
- Application
- libpcap/dump tool

Kernel Space
- System Call Interface
- Protocol Agnostic Interface
- Network Protocols
- Packet Interface
- Device Agnostic Interface
- Device Driver
- Physical Device Hardware

GRO, LRO
➢ When you run tcpdump, it will put your NIC into "promiscuous" mode

host1.test.com: [147715] device em1 entered promiscuous mode

➢ Promiscuous mode makes the Network Card pass all traffic it receives to the central processing unit rather than just frames addressed to it

➢ If a network device is in promiscuous mode, the kernel will receive all network traffic (i.e., the CPU load will slightly increase).
Tcpdump filters

➢ Most basic way of using tcpdump
   
   # tcpdump ( captures everything )

➢ src/dst, port, protocol : combined all three
   
   # tcpdump src port 1025 and tcp
   # tcpdump udp and src port 53

➢ Rotating with timestamps – Every 1 hour
   
   # tcpdump -i eth0 -G 3600 -w 'file.pcap'

➢ Rotating by size – 100MB of data
   
   # tcpdump -i eth0 -C 100 -w capture
How to Get a Good Packet Capture?

- **s0**: Capture the whole packet instead of the first 68 bytes.
- **n**: don't resolve hostnames (faster). Dns lookups can slow down capturing and potentially cause missed packets.
- **i**: listen on a specific interface: `tcpdump -i eth0`, `tcpdump -i bond0`
- **w**: dump packet data to a file, instead of decoding and printing on console
- **host**: Capture only the packets to/from the <host>

```
# tcpdump -s0 -n -i ethX -w /tmp/$(hostname)-$(date +"%Y-%m-%d-%H-%M-%S").pcap host <ip-address>
```

- **s0**: Capture the whole packet instead of the first 68 bytes.
- **n**: don't resolve hostnames (faster). Dns lookups can slow down capturing and potentially cause missed packets.
- **i**: listen on a specific interface: `tcpdump -i eth0`, `tcpdump -i bond0`
- **w**: dump packet data to a file, instead of decoding and printing on console
- **host**: Capture only the packets to/from the <host>

```
#tcpdump -i bond0 -s0 -n -w /tmp/nfsclient.pcap host <ip-nfs-server>
```
Wireshark

- Captures packets and allows you to examine the packet content.
- Provided by the package `wireshark-gnome` in CentOS.
- Wireshark and Time Zones

```bash
# TZ="America/Los_Angeles"  wireshark <file.pcap>
# TZ="Asia/Kolkata"  wireshark <file.pcap>
```

- Run Wireshark with the TZ environment variable set to refer to the preferred time zone.
Tcpdump Analysis
Dynamic Host Configuration Protocol (DHCP)

RFC: https://www.ietf.org/rfc/rfc2131.txt
dhclient.pcap Captured during DHCP IP assignment process
1) Client uses Random Port to connect to Server's Port 21
2) Client sends a PASV command to Server, requesting a port it wishes to use for the Data Channel
3) The server replies with the port number which the client then uses to initiate an exchange on the Data Channel.
Understanding the protocol is essential

NFSv3 : http://www.ietf.org/rfc/rfc1813.txt

NFS Server : 10.70.35.111
NFS Client : 10.70.35.80

NFS Client Capture : tcpdump -i eth0 -s0 -w /tmp/example2.pcap
host 10.70.35.11

[root@nfsclient ~]# mount 10.70.35.111:/test /mnt
[root@nfsclient ~]# cd /mnt/
[root@nfsclient mnt]# touch t1
touch: cannot touch `t1': Permission denied
Getting the following error while restarting network service.

```
host1 ifup:Error, some other host already uses address x.x.x.x
```

Who throws this error? `/etc/sysconfig/network-scripts/ifup-eth`

```
/sbin/arping -q -c 2 -w 3 -D -I ${REALDEVICE} ${ipaddr[$idx]}
if [ $? = 1 ]; then
  net_log "Error, some other host already uses address ${ipaddr[$idx]}."
  exit 1
fi

# tcpdump -i any -w /tmp/tcpdump.pcap arp
```

MAC Address Lookup: IEEE OUI (Organizationally Unique Identifier) and Company_id Assignments

```
```
Feedback
Q and A
Thank you!